DAVID L. WALTZ
Assistant Professor of Electrical Engineering
and Research Assistant Professor
Coordinated Science Laboratory
University of Illinois
Urbana-Champaign, Illinois

Natural-Language Question-Answering Systems

In his excellent book, *Libraries of the Future,*[1] J.C.R. Licklider paints an elaborate picture of what libraries may become by the year 2000. He sees libraries as being accessible through and augmented by digital computer programs and evolving into "procognitive systems," or general aids to thinking. Many library documents, as well as much text, such as that of computer-typeset books have already been made computer-readable. But how far have we come in devising programs that do this reading automatically? And how close are we to systems that can understand users' questions, comments and commands? These are questions I will attempt to answer in this paper.

The systems I will describe all deal primarily with facts rather than documents. I assume that facts are inherently more difficult to deal with, and that documents are a special case of fact.

Since there is a limited amount of space for this presentation, and I wish to put forth some idea of what can be done with current natural-language question-answering systems, I will concentrate on the behavior of the

This work is supported in part by the Office of Naval Research under Contract N00014-67-A-0305 and in part by Joint Services Electronics Program (U. S. Army, U. S. Navy and U. S. Air Force) under Contract DAAB-07-72-C-0259.

systems, and not go into as much detail about how the systems work. I will, however, leave adequate pointers so that those who wish to find more information can do so.

A Brief History

Natural-language technology has advanced dramatically in the last fifteen years. We now have some systems which are not toys, but are in active use by researchers; furthermore, we have a much better idea of what is necessary to generate programs more capable of understanding language in the next generation.

To get an idea of the size of this change, let us first consider BASEBALL, one of the earliest natural-language systems.[2] BASEBALL, written in 1961, answered questions about baseball data comparing month, day, place, teams and scores for each game in the American League for one year. In this limited context, a very small vocabulary was sufficient, since relatively few types of questions could be asked. Furthermore, a user's language was severely constrained. Sentences could contain no dependent clauses, no logical connectives like *and, or,* and *not*), no constructions with relations like *highest* and *most,* and no reference to sequential facts, as in: Did the Red Sox ever win six games *in a row?* Examples of questions BASEBALL could answer include: Who did the Red Sox lose to on July 5? Did every team play at least once in each park in each season? What teams won ten games in July?

BASEBALL operated by parsing its questions, and then transforming the parsed question into a standard "specification list." The question-answering routine took this canonical form as the meaning of the question. Thus "Who did the Red Sox lose to on July 5?" was transformed into the specification list:

Team (losing) = Boston
Date = July 5
Team (winning) = ?

Aside from its grammatical limitations within its domain of expertise, BASEBALL had the following limitations:

1. It could be extended to new domains only by extensive reprogramming.
2. It either understood a sentence fully, or did not understand it at all—no provision was made for saving understood portions of sentences or for interacting with the user to ask clarifying questions.
3. It could not understand pronoun reference.
4. It had no ability to accept declarative information; for example, it was

not possible to add to its data base by telling it "The Red Sox beat the Yankees on July 10."

5. A user could not add procedural information, e.g., one could not add to its linguistic ability, nor give it advice in any form.
6. Because its universe of discourse was so limited, BASEBALL's writers simply never had to worry about handling ambiguous requests.

In contrast, there exist today programs which exhibit—at least to some degree—all abilities mentioned above that BASEBALL lacked. I will briefly describe three programs—those of Woods, Winograd, and Schank—and will then discuss some ideas (principally those of Minsky) which suggest methods for writing vastly more powerful language-understanding programs.

LUNAR

The LUNAR system developed by Woods[3] answers questions about a fairly large data base of samples of lunar rocks and soils. While the data base, like BASEBALL's data base, contains only a small number of data types, the LUNAR system is much more flexible than BASEBALL, most dramatically in its linguistic ability. It is able to accept grammatically complex sentences involving nested dependent clauses, comparative and superlative adjective forms, and some types of anaphoric reference. For example, Woods's system is capable of answering all of the following questions: What is the average concentration of aluminum in high alkalai rocks? Has the mineral analcite been identified in any lunar samples? What are those samples? Give me all model analyses of lunar fines. List the rocks which contain chromite and ulvospinel. What minerals have been identified in lunar samples? What is the concentration of lanthanum in rock S10034? The system was demonstrated at a geology conference only six months after work had begun on the project; the system was then able to answer 78 percent of the questions solicited from the audience. It was judged that the system would have been able to answer 90 percent if the program code had been perfected.

Woods's system uses an augmented transition network (ATN) to parse sentences, and then generates a formal query by patching together the code fragments which represent each phrase in the sentence. Like BASEBALL, then, it attempts to produce something like a specifications list, except that it has a much larger set of lists to choose from and is more clever about concatenating them.

I mention Woods's system especially because his work seems most easily extendable to new domains. In fact, it seems to be quite neatly extendable to document retrieval systems; however, extending it to understand text would definitely be a major task.

SHRDLU

SHRDLU, a natural-language system written by Terry Winograd[4] operates in an environment called the "blocks world." SHRDLU is given an internal description of a scene made up of blocks of different shapes (cubes, bricks, pyramids, etc.) plus a box on a table. The system has a single robot hand under its control, which can pick up one block at a time. A user can give the robot commands to move blocks around (e.g., "pick up a big red block"), ask questions about the scene ("Are there any blue blocks in the box?"), about the robot's actions ("Why did you pick up the green block?"), or give SHRDLU procedural information through declarative sentences ("A steeple is a stack containing two green cubes and a pyramid").

SHRDLU keeps a record of its own goals and subgoals. The following is a typical dialog, assuming that SHRDLU has earlier been told to pick up the red block:

User: Why did you pick up the green block?
SHRDLU: To get rid of it.
User: Why did you do that?
SHRDLU: So I could pick up the red block.
User: Why did you do that?
SHRDLU: Because you asked me to.

Furthermore, the system records declarative information in a form that allows it to use the information either to answer questions or to carry out instructions. Thus, after being told what a steeple is, a user may ask "Are there any steeples on the table now?" which SHRDLU will answer appropriately, using the procedural information to check for an instance of a steeple in the data base. Alternatively, the user may say, "Build a steeple," in which case SHRDLU will do so, using the definition of a steeple to construct a program to carry out the building of the structure.

SHRDLU can carry out a dialog with a user to clairfy sentences. For example, if asked, "Are there any purple pyramids on the red block?" it may in turn ask the user, "Do you mean directly on top of, or supported by?"—unless the answer in both cases is "no."

Finally, as illustrated throughout this section, SHRDLU can handle pronoun and phrase reference (e.g., "Why did you do *that*?"), and it can accept arbitrarily complicated sentence structures (e.g., "Does the shortest thing the tallest pyramid's support supports support anything green?").

SHRDLU does this by interpreting all sentences as procedures (i.e., programs) which are then executed to search its data base, or to run block manipulation programs, or to generate new programs. It uses MICROPLANNER,[5] a programming language designed especially to simplify finding items

which satisfy a goal, like GOAL (?X IN BOX), either by searching the data base for an item ?X which is in the box (pattern-directed data base search), or by calling programs which will change the scene and data base so that there is some item ?X in the box (pattern-directed procedure invocation). MICROPLANNER also contains facilities for automatic backup, so that variables can be assigned tentative values which can later be taken back if they do not work out. Thus, the MICROPLANNER program

```
(GOAL (?X IN BOX))
(GOAL (?X IS-A BLOCK))
(GOAL (COLOR ?X YELLOW))
(GOAL (SUPPORTS ?X ?Y))
(GOAL (?Y IS-A PYRAMID))
  (PICK-UP ?X)
```

represents the English sentence, "Pick up any yellow block in the box which supports a pyramid"; the program will automatically try various values for ?X and ?Y until it either succeeds in satisfying all the goals, or until it has exhausted all possible choices for ?X and ?Y.

Winograd's program was modified to answer questions about weather data, but there are difficulties in extending it. First, either the data base on which SHRDLU operates must be rewritten in a MICROPLANNER form (single-level list structures only), or the outive system would have to be extensively reprogrammed. Thus, it seems unreasonable to use the program on a data base such as one of English text. Secondly, the system seems to support naturally only a single context of discourse. While the data base can obviously be made to include items from any number of contexts, MICROPLANNER has no way of neatly segmenting the data base into coherent pieces. Thus, large-scale data bases cannot be easily supported.

MARGIE

MARGIE[6] (Memory Analysis, Response Generation and Inference on English), developed by Schank, is not a question-answering system, but does shed considerable light on the issues important in automatic text understanding. MARGIE accepts sentences about a wide range of topics and stores them in a semantic-net-like structure called a "conceptual dependency diagram." Conceptual dependency diagrams are intended to be canonical deep structures of the input sentences. These diagrams can then be used to generate inferences from or paraphrases of the input sentence.

If MARGIE is told in its inference mode that "John gave Mary an aspirin," it will produce the following output:

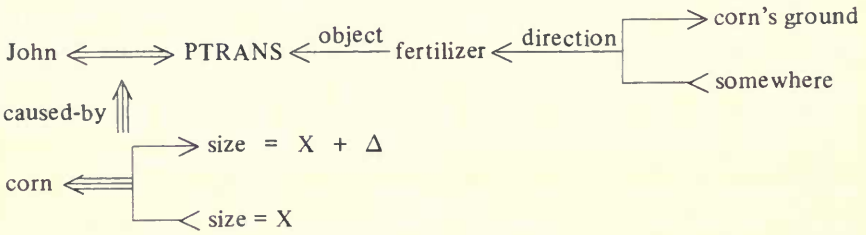
John believes that Mary wants an aspirin.
Mary is sick.
Mary wants to feel better.
Mary will ingest the aspirin.
etc.

If given "John killed Mary by choking Mary," the system in paraphrase mode will produce:

John strangled Mary.
John choked Mary and she died because she could not breathe.
Mary died because she was unable to inhale some air and she was unable to inhale some air because John grabbed her neck.

Conceptual dependency diagrams are formed by replacing all verbs with structures involving twelve basic "ACTS"; Schank argues that these are sufficient to express any verb meaning. The ACTS are: CONC (conceptualize), MTRANS (transfer mental item), ATRANS (transfer possession of article), PTRANS (change location of object), MOVE (body parts), PROPEL (apply a force to), GRASP, INGEST, EXPEL, ATTEND (direct sense organ at something), MBUILD (add information to memory) and SPEAK.

"John grew corn with fertilizer" is represented by the following conceptual dependency diagram:



This can be read as "John caused corn to increase in size by transferring fertilizer from somewhere to the ground around the corn." Thus it can be seen that a substantial amount of structure is attached to the word *grow*. This structure represents the relationships that obtain between the other words in the sentence, using the various symbols in the diagram, each of which has precise meanings.

The structure has slots for other words: *grow* in this case has slots for an agent (John), a plant (corn), an object (fertilizer), as well as slots for instruments (e.g., hoe), which are not filled in this case. Each slot has

associated with it semantic markers which select the types of phrases or words appropriate for the slot. The words which appear in the sentence can be checked against the slots in the verb structure to select the appropriate meaning of the verb. Thus the meaning of *grow* in "John grew the corn" requires a human agent and plant object, whereas the meanings of *grow* in "John grew" or "the corn grew" or "John grew pigs" or "John grew warts" require different types of slot-filling elements. MARGIE uses this slot-filling technique to avoid parsing the sentence in any traditional sense.

Each structure also has links to plausible inferences which can be drawn from the sentences. The system can infer from "John grew the corn" that "John will probably harvest the corn," "John probably wants to either sell the corn or eat the corn," and so on. In a full text-understanding system, these inferences could be used to answer questions not explicitly contained in the text, or to verify the accuracy of its interpretations by comparing the text following a sentence with what the system judges to be appropriate follow-up statements. The system could also be made to understand that a statement such as "John grew the corn and then threw it away" is unusual and requires some additional explanation.

Frame Theory

Minsky has recently written a paper concerning the theory of *frames*, which are semantic structures reminiscent of conceptual dependency diagrams.[7] Like the diagrams, frames have relations, slots, default values and semantic markers, but unlike them, frames may also contain procedural information, and need not correspond only to verbs. Frame theory argues that statements such as "John is a doctor" actually express a complex of plausible inferences simultaneously: John is a physician, John probably has knowledge of anatomy and medication, John may have a specialty, John probably likes to play golf, etc.

Minsky's key idea is that language does not involve the transfer of structures from one speaker to another, as much as it does the selection of a structure in the hearer, and an instantiation of some values in this structure. In this view, listening is an active process, and inherently involves projection (in the psychological sense) on the part of the listener. For example, many jokes are "funny" because the unexpected happens. In any given communication, a number of frames are selected, including a frame for the type of communication (lecture, argument, story, chat, etc.), as well as a frame for topics.

In a large-scale text-understanding and question-answering system, one can imagine frames being selected by the use of keywords, and being verified by matching; this system allows effective segmentation of a system's knowledge into contexts, but also provides links between various contexts. Problems

of ambiguity can often be avoided by knowing the context of the communication; "The group lacked an identity" is only ambiguous if we do not know whether the discourse context is mathematics or psychology. Speed and efficiency can be greatly improved by keeping only the current context in primary storage.

Finally, depth of understanding can be greatly increased through the application of frame theory ideas. Anaolgy and metaphor can be understood as frame transfers; even some poetry and humor may be understandable.[8]

There is every reason to believe that the next generation of language-understanding systems will be as dramatic an improvement over current systems as current systems have been over those of fifteen years ago. While none of the systems described here deals explicitly with current library problems, the descendants of these systems could eventually revolutionize the entire structure of libraries, as well as the lives of all those who use and benefit from libraries.

REFERENCES

1. Licklider, J. C. R. *Libraries of the Future*. Cambridge, Mass., M. I. T. Press, 1965.

2. Green, Bert F., Jr., *et al.* "BASEBALL: An Automatic Question Answerer." *In* Edward A. Feigenbaum and Julian Feldman, eds. *Computers and Thought*. New York, McGraw-Hill, 1963, pp. 207-16.

3. Woods, W. A., *et al.* "The Lunar Sciences Natural Language Information System: Final Report" (BBN Report No. 2378). Cambridge, Mass., Bolt, Beranek and Newman, 1972.

4. Winograd, Terry. *Understanding Natural Language*. New York, Academic Press, 1972.

5. Sussman, G. J., *et al.* "MICRO-PLANNER Reference Manual" (Memo No. 203A). Cambridge, Mass., M. I. T. Artificial Intelligence Lab, 1971.

6. Schank, Roger, *et al.* "MARGIE: Memory, Analysis, Response Generation, and Inference on English." In *Advanced Papers of the Third International Joint Conference on Artificial Intelligence*. Stanford, Calif., Stanford University, 1973, pp. 255-61; and Schank, Roger C. "Identification of Conceptualizations Underlying Natural Language." *In* Roger C. Schank and Kenneth M. Colby, eds. *Computer Models of Thought and Language*. San Francisco, W. H. Freeman, 1973, pp. 187-247.

7. Minsky, Marvin L. "A Framework for Representing Knowledge" (Memo No. 306). Cambridge, Mass., M. I. T. Artificial Intelligence Lab, 1974.

8. Waltz, David L. "On Understanding Poetry." Paper presented at a Conference on Theoretical Issues in Natural Language Processing, June 10-13, 1975, M. I. T., Cambridge, Mass.